SCRUM

lunes, 23 de septiembre de 2024 19:46

- **Transparencia**: Todo el proceso debe ser claro y visible para todos los involucrados. Los aspectos clave del trabajo deben ser comprensibles para aquellos que toman decisiones. Sin transparencia, no se puede inspeccionar de manera adecuada.
- **Inspección**: De manera regular y frecuente, el equipo revisa y evalúa el progreso y los resultados. La inspección ayuda a identificar posibles problemas o desviaciones respecto a los objetivos esperados.
- **Adaptación**: Basado en lo que se aprende a través de la inspección, los procesos o productos deben ajustarse. Si se encuentra que algo no está funcionando, se cambia rápidamente para optimizar el desempeño y los resultados.

Estos pilares del empirismo permiten a los equipos ágiles reaccionar rápidamente a los cambios y aprender continuamente a partir de la experiencia, lo que es clave para mejorar la calidad y la efectividad de los productos o servicios que desarrollan.

Manifiesto Ágil
Promueve un enfoque de trabajo más flexible, colaborativo y adaptativo
Se basa en **cuatro valores fundamentales** y **doce principios** que guían el desarrollo ágil

CUATRO VALORES FUNDAMENTALES

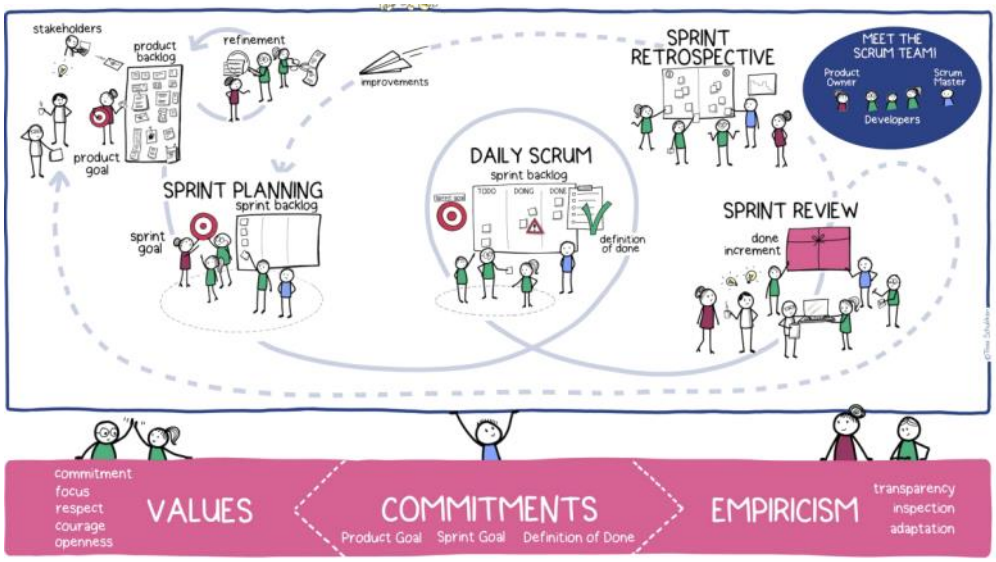
1. **Individuos e interacciones sobre procesos y herramientas**:
   Se da más valor a las personas y la comunicación entre ellas que a los procesos formales o las herramientas tecnológicas. El trabajo en equipo y la colaboración son más importantes que seguir un procedimiento rígido.
2. **Software funcionando sobre documentación extensa**:
   El objetivo principal es crear software que funcione y que aporte valor al cliente, en lugar de dedicar mucho tiempo a generar documentación excesiva. Se prioriza el valor entregado a través de un producto real.
3. **Colaboración con el cliente sobre negociación contractual**:
   Se fomenta la colaboración continua y cercana con el cliente, en lugar de limitarse a cumplir estrictamente un contrato. Se entiende que los requerimientos pueden cambiar y que el cliente debe estar involucrado durante todo el proceso de desarrollo.
4. **Responder al cambio sobre seguir un plan**
   En lugar de aferrarse a un plan predeterminado, los equipos ágiles prefieren ser flexibles y adaptarse a los cambios a medida que surgen, ya que el entorno y las necesidades pueden evolucionar.

LOS 12 PRINCIPIOS DEL MANIFIESTO AGIL

1. Satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptar los cambios en los requisitos, incluso en etapas avanzadas del desarrollo. La agilidad aprovecha los cambios para brindar una ventaja competitiva.
3. Entregar software funcional con frecuencia, desde un par de semanas hasta un par de meses, con preferencia a los períodos más cortos.
4. La colaboración diaria entre desarrolladores y personas del negocio.
5. Motivar a las personas involucradas. Proveerles el entorno y el apoyo necesario, y confiar en ellos para que realicen bien el trabajo.
6. La forma más eficiente y efectiva de comunicar información es mediante la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos ágiles promueven un desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de trabajo.
9. La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
    a. Para ser verdaderamente ágiles, los equipos deben prestar especial atención a la **calidad técnica** del producto y al **buen diseño** del sistema o software que están construyendo.
        i. **Excelencia técnica**: Se refiere a que el código debe ser limpio, bien estructurado, fácil de mantener y escalable. Los desarrolladores deben seguir buenas prácticas, como el uso de pruebas automatizadas, la refactorización de código, la revisión de código y la gestión adecuada de dependencias.
        ii. **Buen diseño**: Un diseño robusto y bien pensado facilita la evolución del software y lo hace más adaptable a cambios futuros. Un mal diseño puede resultar en un sistema difícil de modificar, lleno de "deudas técnicas", lo que obstaculiza la capacidad de ser ágil.
10. La simplicidad—el arte de maximizar la cantidad de trabajo no realizado—es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos autoorganizados.
    a. Lo que quiere decir es que cuando se da a los equipos la libertad de gestionar su propio trabajo y tomar decisiones sobre cómo llevarlo a cabo, suelen surgir soluciones de mayor calidad en términos de **arquitectura, diseño y requisitos**.
12. A intervalos regulares, el equipo reflexiona sobre cómo ser más eficaz, y ajusta su comportamiento en consecuencia.
    - **Reflexión regular**:
      Los equipos deben detenerse periódicamente, generalmente al final de un ciclo de trabajo (como en una **retrospectiva** en Scrum), para evaluar qué ha funcionado bien y qué no. Esto les permite tener una visión crítica de sus procesos, herramientas, dinámicas de equipo y cualquier otro aspecto que afecte su productividad.
    - **Aprendizaje y mejora**:
      A partir de esta reflexión, el equipo identifica oportunidades para mejorar, tanto en los procesos como en la colaboración. Puede tratarse de mejorar la comunicación, hacer cambios en las herramientas que utilizan o modificar sus prácticas de trabajo diario.
    - **Ajustar y perfeccionar**:
      No basta con identificar problemas o áreas de mejora, el equipo debe **actuar** sobre esas conclusiones. Esto implica hacer ajustes tangibles en su forma de trabajar para mejorar continuamente.

SCRUM

Scrum es un marco de trabajo liviano que ayuda a las personas, equipos y organizaciones a generar **valor** a través de soluciones adaptativas para problemas complejos.



1. **Sprint**:
   Ciclo de trabajo con una duración fija (generalmente de 1 a 4 semanas). El Sprint comienza con la Sprint Planning y termina con la Sprint Review y la Sprint Retrospective. Durante el Sprint, el equipo se enfoca en completar un Sprint Goal (objetivo del sprint) y entrega un Increment (un producto funcional).
   a. Sprint Planning
      Reunión que se realiza al inicio de cada Sprint, en la que el equipo Scrum define qué se va a trabajar y cómo se va a lograr.
      Alinear al equipo sobre qué se hará y cómo se logrará durante el Sprint. Es una oportunidad para que el equipo y el **Product Owner** se aseguren de que todos están en sintonía sobre las prioridades y el trabajo que debe realizarse.

      - El equipo revisa el **Product Backlog**, que es una lista priorizada de las tareas o características pendientes del producto, y selecciona los elementos que pueden completarse en el Sprint.
      - Se discute y define el **Sprint Goal**, que es el objetivo del Sprint.

   b. Sprint Review
      La **Sprint Review** es la reunión que se realiza al final de cada Sprint para inspeccionar el trabajo realizado y recibir retroalimentación de los stakeholders.
      ¿Qué sucede en la Sprint Review?
      - El equipo presenta el Increment del producto, que es la parte funcional y completa del trabajo que se realizó durante el Sprint.
      - Los stakeholders (partes interesadas) asisten para revisar lo que se ha completado y dar retroalimentación.
      - El equipo discute qué se completó durante el Sprint y qué quedó pendiente.
      - El Product Owner revisa el Product Backlog para reordenar las prioridades con base en los comentarios recibidos.

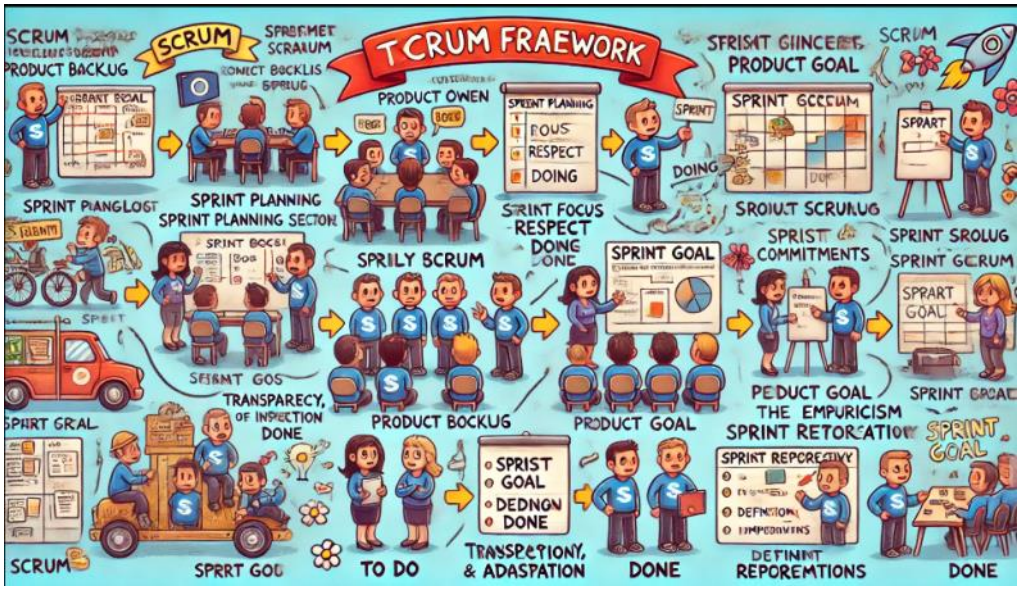
      Mostrar el progreso del producto, obtener retroalimentación y ajustar el Product Backlog para futuras iteraciones. Es un evento clave para asegurar que el producto esté alineado con las necesidades del cliente.
   c. Sprint Retrospective:
      es una reunión que también se lleva a cabo al final de cada Sprint, pero a diferencia de la Sprint Review, aquí el equipo se enfoca en reflexionar sobre cómo han trabajado y cómo pueden mejorar.
   d. Sprint Goal:
      es el objetivo del Sprint, una breve descripción de lo que el equipo planea lograr durante el Sprint. Es un compromiso que orienta al equipo durante el ciclo de trabajo.
      ¿Cómo se define?
      - Se discute durante la Sprint Planning y se basa en los elementos seleccionados del Product Backlog.
      - Proporciona una guía clara que le da al equipo una meta unificadora.

      Aunque los detalles de las tareas individuales pueden cambiar durante el Sprint, el Sprint Goal proporciona una dirección general que guía el trabajo.
   e. **Increment**:
      es el conjunto de todo el trabajo completado durante el Sprint que está listo y cumple con la **Definition of Done** (Definición de Terminado). Es una parte funcional del producto que puede ser entregada al cliente o implementada.
      - ¿Qué características tiene el Increment?
        - Debe ser un software funcional y potencialmente entregable.
        - Cada Increment debe sumar valor al producto general, es decir, debe hacer que el producto esté un paso más cerca de estar completo.
      - Propósito: El Increment es la medida tangible del progreso del equipo. Al final de cada Sprint, el equipo debería tener un Increment que sea usable y que aporte valor al cliente o al producto final.

2. **Eventos de Scrum**:
   a. Sprint Planning: Es la reunión al inicio de cada Sprint. En ella, el equipo selecciona elementos del Product Backlog (lista priorizada de tareas) y los coloca en el Sprint Backlog (tareas para completar en el sprint). Se define el Sprint Goal, que es el objetivo a lograr durante el Sprint.
   b. Daily Scrum: Cada día, el equipo se reúne brevemente (habitualmente 15 minutos) para sincronizar el trabajo. Aquí, se discuten qué se hizo el día anterior, qué se va a hacer hoy y si hay obstáculos. Se revisa el Sprint Backlog para asegurar que todo avanza hacia el Sprint Goal.
   c. Sprint Review: Al final del Sprint, el equipo presenta el Increment (la parte funcional del producto) a los stakeholders (partes interesadas) para recibir retroalimentación.

d. Sprint Retrospective: Después de la Sprint Review, el equipo reflexiona sobre el proceso, identificando lo que funcionó bien y lo que se puede mejorar para futuros sprints. Aquí se hace énfasis en la mejora continua.

3. **Product Backlog y Refinement**:
   Lista priorizada de todas las tareas y características pendientes por desarrollar. Los Stakeholders (partes interesadas) y el Product Owner son los responsables de definir y priorizar estas tareas.
   Durante los sprints, se realiza el Refinement o refinamiento, que es el proceso de mejorar y detallar los elementos del Product Backlog para tenerlos listos para futuros sprints. Es una actividad continua, donde el equipo la sitúa donde le resulte conveniente (Se puede hacer en cualquier comento del tiempo)
4. **Meet the Scrum Team**:
   El equipo Scrum incluye los siguientes roles:
   a. Product Owner: Responsable de maximizar el valor del producto y gestionar el Product Backlog.
   b. Scrum Master: Facilita el proceso Scrum, asegura que el equipo siga las reglas y ayuda a remover impedimentos.
   c. Developers: Son los encargados de transformar los elementos del Product Backlog en un producto funcional.
5. **Valores de Scrum**:
   En la parte inferior de la imagen se mencionan los valores de Scrum:
   - Compromiso (Commitment): El equipo se compromete a lograr el objetivo del Sprint.
   - Enfoque (Focus): Se enfocan en el trabajo del Sprint.
   - Respeto (Respect): Los miembros del equipo se respetan y valoran sus contribuciones.
   - Coraje (Courage): El equipo tiene el coraje para tomar decisiones difíciles y enfrentar los desafíos.
   - Apertura (Openness): Los miembros del equipo son abiertos sobre su trabajo y los desafíos que enfrentan.
6. **Compromisos**:
   En Scrum, hay tres compromisos clave:
   - Product Goal: El objetivo general que el equipo debe alcanzar con el desarrollo del producto.
   - Sprint Goal: El objetivo específico del Sprint.
   - Definition of Done: Los criterios que determinan cuándo un elemento del Sprint Backlog se considera completamente terminado.

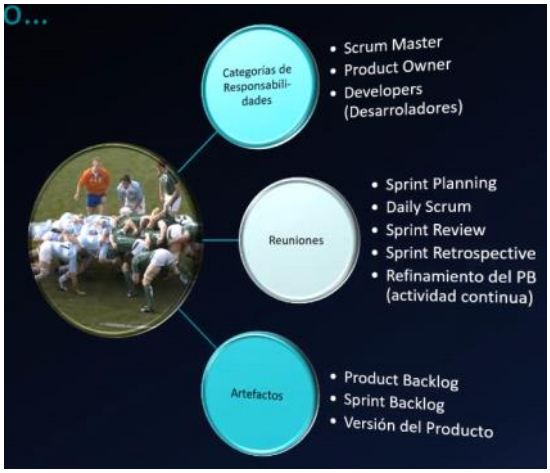


1. **Stakeholders y Product Backlog:**
   Imaginemos que un restaurante quiere una aplicación para que los clientes puedan reservar mesas. El **Product Owner** (el representante del restaurante) trabaja con los **stakeholders** (gerentes del restaurante, equipo de marketing, etc.) para definir las funcionalidades principales que quieren: registrar usuarios, buscar restaurantes, seleccionar horarios, y confirmar reservas.
   - **Product Goal**: Desarrollar una aplicación completa para reservas.
   - **Product Backlog**: Se crea una lista de tareas y características prioritarias, como "permite a los usuarios registrarse", "buscar restaurantes", "elegir mesas", etc.
2. **Refinement del Product Backlog:**
   Durante las semanas previas, el **Product Owner** se reúne con el equipo de desarrollo para aclarar los detalles de estas tareas. Por ejemplo, en la tarea "registro de usuarios", se define cómo será la interfaz y qué datos se necesitarán (nombre, email, contraseña).
   - Se asegura que las funcionalidades más urgentes estén bien especificadas para que el equipo pueda trabajarlas sin problemas e n los próximos sprints.
3. **Sprint Planning:**
   El equipo de desarrollo, el **Scrum Master** y el **Product Owner** se reúnen al inicio del **Sprint** (que durará 2 semanas) para planificar las tareas a realizar. El **Product Owner** prioriza las funcionalidades más urgentes, como el registro de usuarios y la búsqueda de restaurantes.
   - **Sprint Goal**: Permitir que los usuarios se registren en la aplicación.
   - **Sprint Backlog**: Se eligen tareas como "diseñar la pantalla de registro", "implementar la base de datos de usuarios", "crear validación de datos".
4. **Sprint:**
   El Sprint comienza. Durante estas dos semanas, el equipo trabaja en las tareas que seleccionaron. Desarrollan la pantalla de registro, validan los datos que los usuarios ingresan y configuran la base de datos.
   - Cada miembro tiene tareas asignadas, y todos colaboran para cumplir el **Sprint Goal** (que los usuarios puedan registrarse).
5. **Daily Scrum:**

Cada día, el equipo se reúne por 15 minutos en el **Daily Scrum**. Aquí, cada desarrollador comenta lo que hizo el día anterior, lo que va a hacer hoy, y si enfrenta algún bloqueo.
    - Por ejemplo, uno de los desarrolladores menciona que no puede conectarse a la base de datos, y otro compañero lo ayuda a resolverlo.
6. **Sprint Review:**
   Después de 2 semanas, el Sprint finaliza, y el equipo realiza una **Sprint Review**. Aquí, muestran a los **stakeholders** (dueños del restaurante) lo que han completado: la funcionalidad de registro de usuarios está lista.
    - Los **stakeholders** prueban la aplicación, dan sus comentarios y el **Product Owner** decide si es necesario ajustar algo o continuar con la siguiente funcionalidad prioritaria, como la búsqueda de restaurantes.
7. **Sprint Retrospective:**
   Luego, el equipo realiza la **Sprint Retrospective**. Reflexionan sobre el proceso de trabajo del Sprint. Por ejemplo, se dan cuenta de que hubo problemas con la comunicación sobre cómo debía funcionar la validación de datos, así que deciden mejorar la forma en que discuten las especificaciones.
    - También acuerdan realizar sesiones de revisión técnica a mitad del Sprint para prevenir problemas futuros.
8. **Increment:**
   El **Increment** al final de este Sprint es la funcionalidad de registro de usuarios totalmente funcional y lista para ser utilizada en la aplicación.

Ciclo Repetitivo:
- Este proceso se repetirá para cada funcionalidad del **Product Backlog** hasta que el producto esté completo.
- En el siguiente Sprint, por ejemplo, el equipo podría trabajar en la funcionalidad de "búsqueda de restaurantes", con su propio **Sprint Goal** y tareas en el **Sprint Backlog**.

RESUMIENDO



- Categorías de Responsabilidades.
    - No habla de roles, si no de categorías y responsabilidades
    - Una persona puede cambiar de responsabilidades y categoría a la otra
- Reuniones
- <u>Artefactos</u>
    - Cada artefacto en Scrum tiene asociado un compromiso.
        - Este compromiso en el Product Backlog, se materializa como objetivo del producto
            - Osea que vamos a construir un producto que tiene un objetivo concreto
                - Esto nos permite que a medida que vamos agregando Feature a nuestro Product Backlog, nosotros sabemos que tiene que estar enmarcada a este compromiso / objetivo del producto que esta definido
                - Esta definición del Objetivo del producto, nos da una cuota, de flexibilidad. Ya que sabemos que el product Backlog va cambiando en el tiempo. Entonces esta bueno tener identificado este compromiso, para saber que a ese lugar vamos y que eso es lo que no va a cambiar
        - Con el Sprind Backlog pasa algo parecido, se decine el objetivo del sprind y nosotros de alguna manera podemos mevernos dentro del sprind siempre que se siga respetando con el objetivo
        - El incremento del producto que esta asociado, con el compromiso completo es el Definition of Done.

Timebox en Scrum

- Restricción de tiempo máxima establecida para completar una actividad o evento específico. Es un límite de tiempo fijo, inmutable, y se utiliza para asegurar que las actividades se realicen de manera eficiente, sin alargarse innecesariamente.
- También es aprender a negociar en otros términos, que la variable no sea siempre en tiempo. Puede que sera el alcance
- Todas al Ceremonias son TIMEBOX
- Acá no nos tenemos que quedar con la cantidad de Horas, si no que se fije con el equipo y que se respete en el TimeBox.

Sprint: 1 mes o menos
Sprint Planning: 8 horas máximo para un Sprint de 1 mes
Daily Meeting: 15 minutos
Sprint Review: 4 horas máximo para un Sprint de 1 mes
Sprint Retrospective: 3 horas máximo para un Sprint de 1 mes
Refinamiento del PB: 10% del tiempo de Sprint

Recordando lo que es el Definition of Ready y Definition of Done

Ayudan a asegurar que el trabajo sea gestionado de manera eficiente y con calidad. Aunque ambos conceptos están relacionados con el proceso de trabajo, se aplican en diferentes momentos y tienen enfoques distintos.

- **Definition of Ready** (DoR)
  Conjunto de criterios que definen cuándo una tarea (o historia de usuario) está lista para comenzar a trabajar (Entra al Sprind Backlog o no). En otras palabras, establece que el trabajo está suficientemente claro y detallado para que el equipo de desarrollo pueda iniciarlo sin dudas o bloqueos importantes.
  - Tiene que cumplir con el INVEST MODEL

Definición de "Listo" (Ready)	
❑	Valor de negocio claramente expresada.
❑	Detalles suficientemente comprendidos por el Equipo de forma tal que puedan tomar una decisión informada sobre si pueden completar **el ítem del product Backlog (PBI).**
❑	Dependencias identificadas y no hay dependencias externas que puedan impedir que el PBI se complete.
❑	El equipo ha sido asignado adecuadamente para completar el PBI.
❑	El PBI ha sido debidamente estimado y es lo suficientemente pequeño para ser completado en un Sprint.
❑	Los criterios de aceptación son claros y testeables.
❑	Los criterios de performance si hay, son claros y testeables.
❑	El equipo comprende como mostrar el PBI en la Sprint Review.

  - Ejemplo de Definition of Ready:
    Antes de comenzar una tarea o historia de usuario, esta debe cumplir con ciertos criterios, como:
    - La historia está claramente descrita y entendida por todos.
    - Los criterios de aceptación están definidos.
    - Todas las dependencias (otros equipos, herramientas, etc.) han sido resueltas.
    - El equipo ha estimado la tarea (story points o tiempo).
    - Cualquier riesgo o impedimento potencial ha sido identificado.

- **Definition of Done** (DoD)
  Conjunto de criterios que define cuándo una tarea o una funcionalidad se considera completamente terminada para entrar a la Sprind Review, esto para ver si se lo podemos mostrar al cliente. Una vez aceptado nos dice si va a producción
  Es una lista de criterios que el equipo debe cumplir antes de que el trabajo pueda considerarse como "hecho" y listo para ser entregado o lanzado.

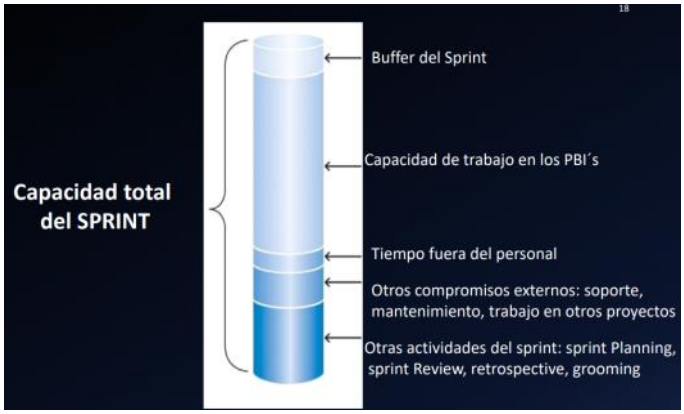
Definición de Hecho (DONE)	
❑	Diseño revisado
❑	Código Completo
❑	Código refactorizado
❑	Código con formato estándar
❑	Código Comentado
❑	Código en el repositorio
❑	Código Inspeccionado
❑	Documentación de Usuario actualizada
❑	Probado
❑	Prueba de unidad hecha
❑	Prueba de integración hecha
❑	Prueba de sistema hecha
❑	Cero defectos conocidos
❑	Prueba de Aceptación realizada
❑	En los servidores de producción

- Los criterios están directamente relacionados con la calidad del producto.
- Ejemplo de Definition of Done:
  Una tarea o historia de usuario debe cumplir con ciertos criterios antes de ser considerada "hecha", como:
  - El código ha sido desarrollado y revisado.
  - Las pruebas unitarias han sido implementadas y pasadas.
  - La funcionalidad ha sido probada y validada por el equipo de QA.
  - La documentación ha sido actualizada si es necesario.
  - El incremento ha sido integrado en el código base y está listo para ser desplegado.

Definition of Ready y Definition of Done:

Aspecto	Definition of Ready (DoR)	Definition of Done (DoD)
Cuándo se aplica	Antes de que una tarea se comience a trabajar (en la planificación).	Al finalizar una tarea o historia de usuario, antes de marcarla como "completada".
Propósito	Garantiza que el trabajo esté claro y preparado para ser abordado.	Garantiza que el trabajo cumpla con los estándares de calidad y esté completamente terminado.
Criterios	Claridad, dependencias resueltas, criterios de aceptación definidos.	Pruebas completas, código revisado, documentación actualizada, funcionalidad validada.

Capacidad del equipo en un Sprint.



La Capacidad es una métrica que se usa para ver cuanto compromiso de trabajo el equipo puede asumir en un determinado Sprint

- Una de las cosas que se hace en el Sprint planning es determinar la capacidad del equipo. Entonces contra esa capacidad se contrasta cuantas User del producto Backlog pueden pasar al Sprint Backlog y hasta cuando nos podemos comprometer para determinado Sprint.
- Porque se Estima ? Se estima porque estamos empezando en el Sprint de la Planificación y estamos viendo cada uno de nosotros cuantas horas podemos dedicarle de Sprint.
- La capacidad se puede estimar en Horas Ideales o en Punto de Historia.
  - Si los equipos son más maduros los equipos pueden estimar en Story Points
- Se analiza cada miembro a cada equipo, cada día disponible para trabajar, las horas por dia que puede dedicar y entonces se puede hacer una determinación de Capacidad, que la recomendación es trabajar por Rango de Tiempos

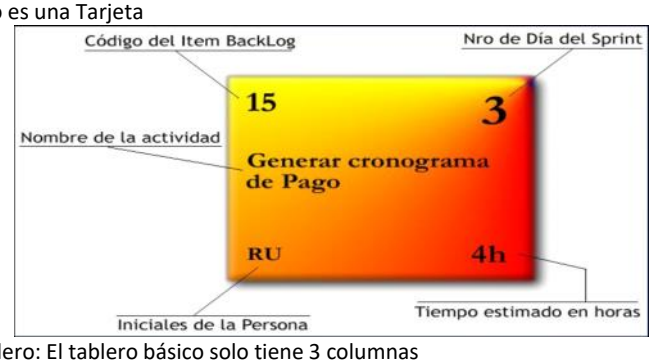
Persona	Días disponibles (sin tiempo personal)	Días para otras actividades Scrum	Horas por día	Horas de Esfuerzo disponibles
Jorge	10	2	4-7	32-56
Betty	8	2	5-6	30-36
Simón	8	2	4-6	24-36
Pedro	9	2	2-3	14-21
Raúl	10	2	5-6	40-48
Total				140-197

- Llegamos a tener un calculo de horas que el equipo tiene disponible.
- Esto va a aplicado al triangulo de hierro.

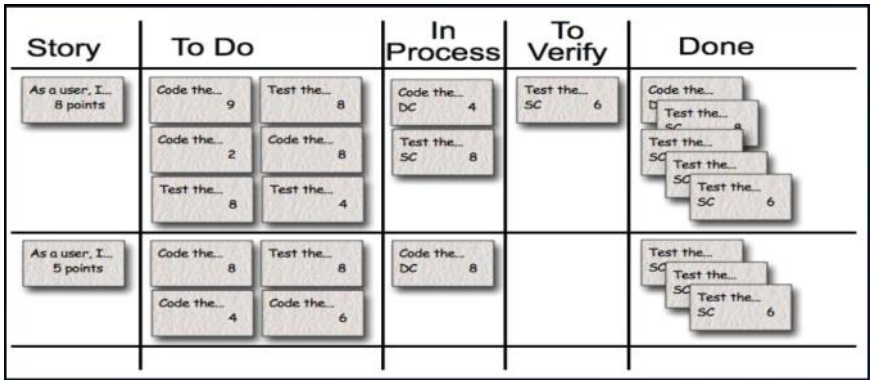
- El tiempo es el Sprint que dura las semanas y esta fijo y por el otro lado los recursos que se refiere a la capacidad del equipo que se calcula y es fija. Y que podemos entregar con esto ? Ahí se define el Alcance
- Entonces:
  - Primera variable de Definición cuanto va a durar el sprint (Lo decide el equipo).
  - Segunda variable de definición, el cálculo de capacidad
  - Despues se acuerdael objetivo del sprint
- Con estas 3 variables empezamos a decir que cosas van a pasar y hasta donde del product backlog al sprint backlog

Herramientas de Scrum

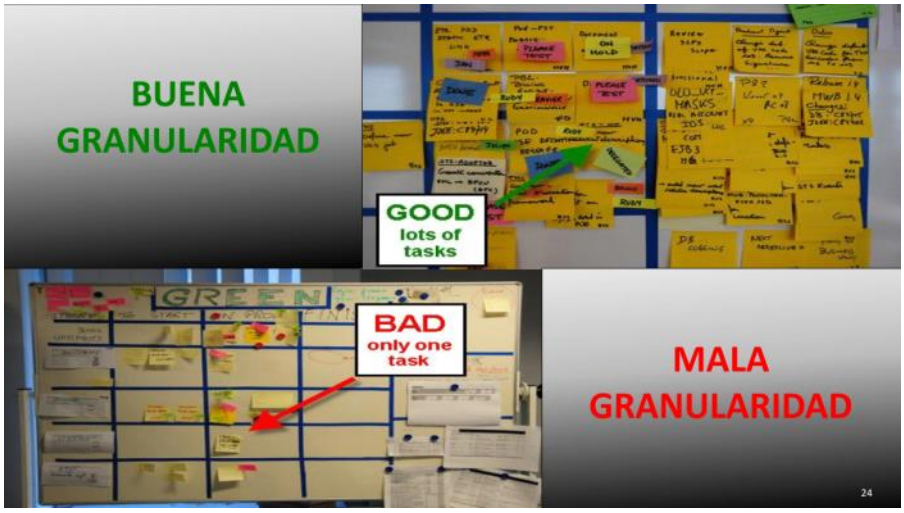
1. **Taskboard**
   a. Esto es una Tarjeta



   b. Tablero: El tablero básico solo tiene 3 columnas



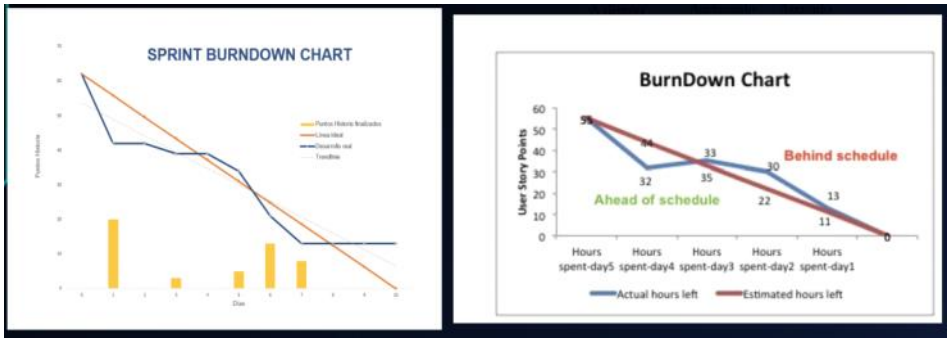
   i. Story: Historias Comprometidas
   ii. To DO: Son las características que salieron del product backlog y entraron al sprint backlog
      1) Cuando haces una desagregación de historias, el equipo divide la esa historia en varias tareas.
      2) Las tareas se estiman en Horas Ideales
   iii. In Process: Básicamente que esta en proceso y se pone las iniciales de la persona que lo esta haciendo
   iv. To Verify : Esta verificada
   v. Done: Cuando la tarea esta terminada



- Granularidad Gruesa: Cuando tiene muchas cosas adentro unas sola User
- Granularidad Fina: Cuando hace algo más concreto.

- Buena Granularidad:
  Acá la Granularidad es más Fina, estamos trabajando con más Historias. Esto nos da la posibilidad de poder maniobrar, asignar , empezarlas y terminarlas antes que se termine el Sprint
- Mala Granularidad:
  Cuando tenemos poquitas tareas (Granularidad Gruesa), es porque son tareas Gordas. Tareas que vienen de una User estimada con puntos de Historia más allá de 8.
- SCRUM recomienda gestión Binaria, como esta en la buena granularidad, con Users Chiquitas.

2. **Sprint Burndown Charts**



Seguimiento del progreso del equipo durante un Sprint. Estas gráficas muestran cuánta **cantidad de trabajo** (medida en horas, puntos de historia, o cualquier otra métrica) queda por hacer en el Sprint, comparado con el tiempo restante hasta el final del Sprint.

Elementos Clave del Sprint Burndown Chart:

a. Eje X (Horizontal):
   i. Representa el tiempo del Sprint, normalmente medido en días. Va desde el día 1 del Sprint hasta el último día (por ejemplo, el día 14 si el Sprint dura dos semanas).
b. Eje Y (Vertical):
   i. Representa la cantidad de trabajo restante por completar. Puede medirse en horas de trabajo, puntos de historia (story points), tareas o cualquier otra métrica utilizada por el equipo.
c. Línea Ideal:
   i. Es una línea descendente que muestra el progreso ideal o esperado. En un mundo perfecto, el equipo completaría una cantidad de trabajo constante todos los días, y al final del Sprint el trabajo restante sería cero.
d. Línea Real (Burndown Real):
   i. Es la línea que muestra el progreso real del equipo. A medida que el equipo completa tareas y reduce el trabajo pendiente, esta línea también debería descender, aunque de manera menos uniforme que la línea ideal.
   ii. Si hay picos que suben, quiere decir que hay reestimación. Ósea hay un nivel de complejidad que se paso
   Ósea una User que era 2 y termino siendo 5.

Cómo Funciona:

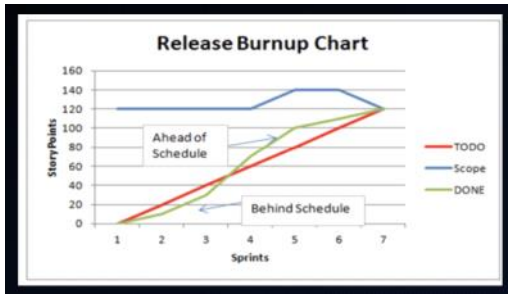
- Al principio del Sprint, el equipo tiene una **cantidad total de trabajo** planificada (por ejemplo, 100 puntos de historia o 150 horas de trabajo).
- Cada día del Sprint (daily), el equipo actualiza la gráfica, indicando cuánto trabajo queda por hacer. A medida que completan tareas, la cantidad de trabajo pendiente disminuye, y la línea en la gráfica va descendiendo.
- Al final del Sprint, la gráfica idealmente debería mostrar que **todo el trabajo ha sido completado** y que la cantidad de trabajo pendiente es **cero**. Termina el Sprint y se borra

Interpretación del Sprint Burndown Chart:

- **Línea descendente constante**: Indica que el equipo está progresando de manera estable y que es probable que terminen el trabajo a tiempo.
- **Línea plana** o con muy poca pendiente: Esto podría significar que el equipo no está avanzando como se esperaba y que podrían tener dificultades para completar todo el trabajo planificado dentro del Sprint.
- **Línea con picos o saltos**: Indica un progreso irregular, lo que puede ser normal en algunos casos, pero si es muy frecuente, podría

señalar problemas en la estimación o ejecución del trabajo.

3. **Gráficos de Seguimiento del Producto**
   Seguimiento del progreso del equipo en el Producto, muestran cuánto trabajo se ha completado a lo largo del tiempo y también pueden indicar el total de trabajo a realizar



Componentes Clave del Sprint Burnup Chart:

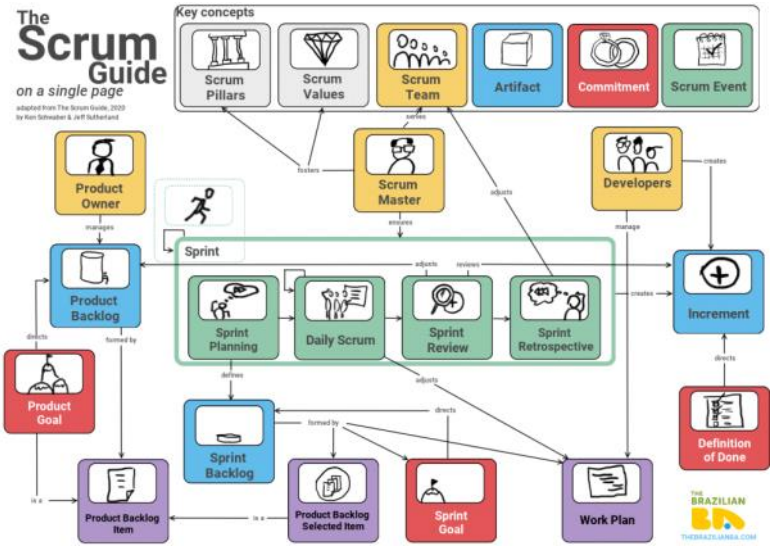
- Eje X (Horizontal):
  - Representa el tiempo del Sprint o proyecto, generalmente en días o Sprints sucesivos.
- Eje Y (Vertical):
  - Representa la cantidad de trabajo completado, que puede medirse en puntos de historia, tareas, horas de trabajo, etc.
- Línea de Trabajo Completado:
  - Muestra el progreso del equipo en cuanto a trabajo completado. Esta línea comienza en cero y va subiendo a medida que el equipo completa historias de usuario.
- Línea de Alcance Total (Opcional):
  - Esta línea representa el alcance total del trabajo (el total de puntos de historia o tareas planificadas para el proyecto). Si el alcance cambia durante el Sprint (por ejemplo, si se agregan o eliminan tareas), esta línea también se ajusta, lo que hace visible cualquier cambio en el trabajo planificado.

Cómo Funciona:

- Al inicio del Proyecto, el equipo tiene una cantidad total de trabajo planeado, y a medida que completan tareas, la línea de trabajo completado comienza a subir.
- El objetivo es que la línea de trabajo completado llegue al mismo nivel que la línea de alcance total al final del Proyecto, lo que indica que todo el trabajo planeado ha sido completado.

Beneficios de las Burnup Charts:

a. Visibilidad del Progreso y del Cambio de Alcance:
   i. A diferencia de las burndown charts, las burnup charts muestran no solo el progreso del equipo, sino también cambios en el alcance. Si se agregan más tareas al Sprint, la línea de alcance sube, lo que deja claro que ha habido un cambio en la cantidad de trabajo a realizar.
b. Mejor Visualización de Riesgos:
   i. Facilitan la identificación de problemas no solo cuando el equipo no progresa lo suficiente, sino también cuando el alcance del trabajo se incrementa. Es más fácil ver si el equipo está completando el trabajo a tiempo o si hay problemas de sobrecarga.
c. Monitoreo del Avance hacia el Objetivo:
   i. Permiten ver cuánto progreso se ha hecho y cuánto trabajo queda, pero de manera acumulativa, lo que ayuda a visualizar si el equipo está en camino de completar todo el trabajo.



1. **Product Owner (Propietario del Producto)**
   El Product Owner en este caso es alguien del área de negocio que quiere desarrollar una app para pedir comida a domicilio. Su responsabilidad es definir lo que debe tener la aplicación y en qué orden se debe construir. Por ejemplo:
   - Características necesarias: Registro de usuarios, menú de restaurantes, carrito de compras, integración de pagos, seguimiento del pedido.

2. **Product Backlog (Lista de Producto)**

El **Product Backlog** es la lista de todas las funcionalidades que necesita la aplicación, priorizadas según su importancia. Un ejemplo de elementos del Product Backlog podría ser:

- Registro de usuarios.
- Búsqueda de restaurantes.
- Carrito de compras.
- Pasarela de pagos.
- Seguimiento del pedido.

El Product Owner gestiona esta lista y la va actualizando a medida que surgen nuevas ideas o se ajustan las prioridades.

3. **Product Goal (Objetivo del Producto)**
   El **objetivo del producto** sería, por ejemplo: "Crear una aplicación que permita a los usuarios pedir comida a domicilio de manera rápida y sencilla". Este objetivo es la visión a largo plazo.

4. **Sprint**
   El desarrollo se organiza en **Sprints**, que son ciclos de trabajo de 2 semanas. En cada Sprint, el equipo selecciona las funcionalidades más importantes para completar dentro de ese tiempo.

5. **Sprint Planning (Planificación del Sprint)**
   En la reunión de planificación del Sprint, el equipo decide qué funcionalidades del **Product Backlog** van a desarrollar en este Sprint. Supongamos que el objetivo de este Sprint es permitir a los usuarios **registrarse y navegar por los restaurantes**. Así que seleccionan esos ítems del **Product Backlog** y los añaden al **Sprint Backlog**.

6. **Sprint Backlog (Lista de Tareas del Sprint)**
   El equipo selecciona las siguientes tareas del Product Backlog y las coloca en el **Sprint Backlog**:
   - Implementar el formulario de registro de usuarios.
   - Crear la pantalla principal con la lista de restaurantes.
   - Conectar la pantalla principal con la base de datos de restaurantes.

   El equipo sabe que estas tareas deben completarse en las próximas dos semanas y que todo lo que se haga debe cumplir el objetivo del Sprint.

7. **Daily Scrum (Reunión Diaria)**
   Cada día, el equipo de desarrolladores se reúne para el **Daily Scrum**, una breve reunión de 15 minutos donde cada persona explica qué hizo el día anterior, qué va a hacer hoy y si tiene algún obstáculo. Por ejemplo:
   - "Ayer completé el diseño del formulario de registro. Hoy trabajaré en conectar la pantalla de restaurantes con la base de dat os."

8. **Sprint Review (Revisión del Sprint)**
   Al final de las dos semanas, el equipo realiza una **Sprint Review** donde muestran al Product Owner y a otros interesados lo que han completado:
   - Los usuarios pueden registrarse en la aplicación.
   - Pueden ver una lista de restaurantes disponibles.

   El Product Owner y los stakeholders dan su feedback. Si todo está bien, las funcionalidades se consideran completas.

9. **Sprint Retrospective (Retrospectiva del Sprint)**
   Después de la revisión, el equipo realiza una retrospectiva para analizar cómo les fue en el Sprint. Identifican cosas que salieron bien (ej., buena colaboración en el equipo) y cosas que pueden mejorar (ej., mejorar la velocidad de desarrollo).

10. **Developers (Desarrolladores)**
    El equipo de desarrollo es responsable de transformar los ítems del **Sprint Backlog** en un **Incremento** del producto. En este Sprint, eso significaba desarrollar el registro de usuarios y la pantalla de restaurantes.

11. **Increment (Incremento)**
    El **Incremento** es el resultado del Sprint: una versión de la aplicación que permite a los usuarios registrarse y ver restaurantes. Este incremento es funcional y cumple con los criterios de la **Definition of Done**.

12. **Definition of Done (Definición de Hecho)**
    La **Definition of Done** es un acuerdo que define cuándo una tarea está completamente terminada. En este caso, podría ser que una funcionalidad está "hecha" cuando:
    - Ha sido programada.
    - Ha sido probada (sin errores).
    - Cumple con los requisitos del Product Owner.
    - Está lista para ser usada por los usuarios.

13. **Scrum Master**
    El **Scrum Master** se asegura de que el equipo siga los principios de Scrum y que no haya impedimentos en su trabajo. Si los desarrolladores tienen problemas, el Scrum Master ayuda a resolverlos para que puedan avanzar.

14. **Scrum Team (Equipo Scrum)**
    El **Scrum Team** está compuesto por el **Product Owner**, los **Developers** y el **Scrum Master**. Juntos trabajan para alcanzar los objetivos de cada Sprint y finalmente cumplir el **Product Goal**.

Flujo del Ejemplo:

1. **Product Owner** define la visión del producto: crear una app de comida a domicilio.
2. Se crea el **Product Backlog** con todas las características necesarias.
3. Se establece el **Product Goal**: "Permitir a los usuarios pedir comida fácilmente".
4. Cada Sprint, el equipo selecciona tareas del **Product Backlog** y las desarrolla.
5. Durante el Sprint, el equipo se sincroniza en reuniones diarias.
6. Al final del Sprint, presentan el **Incremento**: nuevas funcionalidades listas para usarse.
7. El ciclo continúa con la planificación del siguiente Sprint hasta completar la aplicación.